Department of Mathematics and Computer Science, Gordon College

Format for Final Submission of Computer Science Senior Projects (rev 7/14)

The formal submission of a senior project to the department will consist of a binder containing the following documents, in the order listed, together with machine-readable source code for the project (on a CD or departmental server). A preliminary submission is due on the Monday following spring break, with the final submission being due near the end of spring semester, as shown in the syllabus. A copy of the "Evaluation Form for Senior Project Notebook" (Appendix D) should be included with the preliminary submission. See discussion of the individual documents below for details regarding the content of each document.

The documentation and approval process outlined here is fairly structured. Of course, often "real-world" projects use a much more flexible approach to development documentation. With smaller projects, in particular, it is common to use a more "agile" or "low ceremony" approach. The main reason for using a more structured approach here is pedagogical: to give you experience with the various issues that need to be considered when doing a project (even if not as formally documented). (Copies of the notebooks from the previous year are available from the professor, and can be borrowed (for a short period of time). You will probably find these very helpful as examples.)

The complete notebook will contain the following, in this order:

1. Approved senior project proposal and problem statement, with all necessary signatures showing acceptance of the original proposal and of the finished project.
2. Software Project Management Plan
3. Social Impact Statement.
4. Requirements Analysis Document.
5. User's Manual. (If your user documentation is structured as one or more web pages, then printouts of those pages should go here.)
6. System Design.
7. Software installation and maintenance documentation, including:
   • An inventory of files needed to build the software, indicating where these files may be found.
   • Commands needed to build the software.
   • Installation procedure for the software.
8. Detailed design documents and source code listings of all software components. This should be preceded by an outline and/or other documentation showing the relationship of the components to one another.
9. Test plan and test report prepared by peer tester. (Final submission only)

Each document must be neatly printed and punched for insertion into the binder. Each of documents (except the proposal - #1) must have a title page with the signature of the client and/or departmental representative showing acceptance of the final form of the document (including changes requested following the original submission.) (See Appendix B for a sample title page.)

- All documents must have the signature of the departmental representative
- Documents 1, 2, 3, 4, 5 must also have the signature of the client. (#1 in two places)

It should be noted that this submission is for the department. You must also give appropriate documentation to the person for whom the project is being done and/or to a technical person who can assist the client with ongoing maintenance of the software. You will probably also want to keep a copy of the notebook for yourself.

Exceptions to this format may be made where appropriate, but must be approved in advance by the departmental representative overseeing the project.

Specific Information about Document Formats

You will probably find it helpful to look at senior projects from recent years to get some idea of what the various documents should look like.

In some cases, there are standard outlines for the document which your document must follow; in other cases, you should create an outline that is appropriate for your project - don't slavishly imitate the examples in the text book or the work of previous years' project teams. Note, though, that the document templates in the Bruegge book are somewhat different from those in other books, so the outline of your documents will not be identical to those done by other students. Where a document template in Bruegge is specified. follow the outline exactly as given. If a section in the outline does not apply, simply indicate "not applicable" rather than skipping it.

Because the document templates are designed to allow each document to be read as a standalone document, there is considerable redundancy between documents, especially in the introductory sections. You should feel free to make use of "cut and paste" (with suitable editing) for these sections.

The following are specific expectations for the various documents:

Senior Project Proposal and Problem Statement

You must use the standard project proposal form (see Appendix A.) All blanks on the proposal form (except for signatures) should be word-processed, not hand-written. A blank copy of the proposal form which can be filled in using a word-processor is available on the course web page.

The problem statement should follow the outline on page 595 of the text by Bruegge - just points 1, 2. At this point, your scenarios are only need to cover the major ways in which a primary actor would make use of your system - you do not need to consider "maintenance" tasks. (See page 47 for an example of a scenario)

Software Project Management Plan (Often called an SPMP)

Use the outline on page 591 of the text by Bruegge, which is simplified from IEEE Standard IEEE 1058.1-1987.

Social Impact Statement (Often called an SIS)

There is no standard format for this document, but it should cover the issues discussed in Shneiderman §3.8.

Requirements Analysis Document (Sometimes called a Software Requirements Specification - SRS)

This document should follow the outline on page 152 of the text by Bruegge.

Bruegge's outline is adapted from IEEE Standard IEEE 830-1993, particularly to incorporate use cases and various UML diagrams. A lot of work needs to go into section 3.4 "System Models" - but investing the effort now will pay off later! You can be a bit flexible with the format of this particular section. Also, remember that this is an analysis document, not a design document, so you should restrict yourself to material that is relevant to understanding what your system is to do, without spelling out how it is to do it. Your document should include a use case diagram in section 3.4.2, along with a scenario for each use case. It may include an analysis class diagram in section 3.4.3, and may include state diagrams, activity diagrams, and/or sequence diagrams (for each use case) in section 3.4.4 if they are helpul in understanding the use case. However, do not feel compelled to include diagrams in §3.4.3 or 3.4.4 if they are not needed for making the requirements clear. If your system is not strictly object-oriented, you may find it useful to include a data-flow diagram in this section - see discussion of DFD's in an earlier book by Schach (borrow from professor) §11.3.

Note the expectation that the document also incorporate a high-level design of the user interface in section 3.4.5. The functional design work you did developing your rapid prototype (i.e. screens flow between screens) should be incorporated here - the rapid prototype itself is not included in the project notebook.

User's Manual

There are no specific format expectations for this document. Your task is to produce a document that would provide a user of your system with the information he/she needs to use it. Put yourself in the user's shoes and ask "what would I want to know if I were using this software for the first time?" (If your project has multiple categories of users, then you may want to create separate manuals for the different categories of users.) Remember that user manuals are normally written in the second person (if you want to ... then ...), whereas specifications are normally written in the third person (the user will be able to ...). The following are some specific areas your manual should cover - not necessarily in the order listed below:

- Overview of the software: Answer the question "What will this software do for me?"
- Discussion of how to start the software, including any required logon process.
- Discussion of how to perform various typical tasks that the user may wish to perform.
- Description of all menus, commands, and other features. (This can be in a reference manual section of the document.)
- Description of error messages the user is likely to see - what they mean and what to do about them.

Note that this document is developed in three stages, rather than two as for the other documents:

- A preliminary version of this document is turned in at the same time as the Requirements Analysis Document. The purpose of creating a preliminary version at this time is to further clarify the functionality and user interface of the proposed system. Although you will not be able to include actual screen shots in this version, you should include simulated screen

shots (prepared with a suitable drawing tool) in the places where the actual screen shots will go later. (Figuring out what the user should actually see on the screen at various points is an important part of designing the system. Do not omit screen shots.)

- An intermediate version is turned in as shown in the course syllabus. The purpose of creating a version at this time is to guide the process of software development by clearly spelling out what the software you are developing is to do. This version should be complete in every way, except for using simulated (rather than actual) screen shots. It is not a draft. It will be graded.
- A final version is turned in with the project notebook. This version should differ from the intermediate version only in including actual screen shots based on the completed software, as well as reflecting any changes made during implementation or testing.

System Design Document

Use the outline on page 285 of the text by Bruegge. Also see Appendix C - "Some Suggestions Regarding the Senior Project System Design Document.". Note that the template in Bruegge is fairly high-level, which is appropriate for a really large system. For your projects, developing the design in more detail (e.g. through the use of class diagram(s), state diagrams, activity diagrams, and/or sequence diagrams) is helpful here. The precise form this document takes will be heavily dependent on the architecture of the system.

- It should identify the major modules comprising the system (classes, scripts, or whatever). For each major module, the design should spell out its purpose (what are its responsibilities?) and its interface (what services does it make available to other modules and what do those other modules have to provide by way of parameters to use those services?)
- If the modules of the system lend themselves to being logically grouped into packages, then it should describe the package structure (perhaps using a package diagram.)
- If the system makes use of any major frameworks or design patterns, then these should be referenced in the design. (The task here would not be to describe the pattern, but rather to discuss how the system will make use of it.)
- If the system makes use of a database, then the system design should include the design of the database, utilizing an ER diagram and/or the schema of the various tables, including specification of primary and foreign keys.
- In any case, the rationale for key design decisions (as discussed in Bruegge ch. 12) should be discussed in the document.

You will have successfully accomplished this task if you produce a document which is such that someone else could build your project from your design by constructing the various modules as specified by the design. Again, working hard now will pay off later!

Software Installation and Maintenance Documentation

There are no specific format expectations for this document. In developing this document, put yourself in the shoes of someone who is installing your software after you have graduated or who comes along several years from now and needs to make a small change to your system. What would this person need to know in order to be able to successfully rebuild/reinstall your system from the sources after making the necessary changes?

Detailed Design Documents and Source Code Listings

To the extent possible, detailed design documentation should be incorporated into the source code in the form of appropriate comments - e.g.

- File/class prologue comments incorporating a statement of purpose and (where appropriate) class invariant information.
- Method prologue comments incorporating a description of parameters and return value and, where appropriate, pre and post-conditions for the method.
- Descriptions of key variables (at the point of declaration if using a language that requires variables to be declared - else at the start of each chunk of code.)
- Comments embedded in the code describing anything that would not be obvious to the reader.

Where appropriate, this should be supplemented with appropriate external documentation such as more detailed class diagrams (showing methods and variables), state, activity, or sequence diagrams, etc. If the code is broken up into packages, then the notebook should contain a description of the purpose of each package, as well as a package diagram (if one was not included in the system design.)

A guiding principle is to put yourself into the shoes of a later maintainer, who will need to be able to understand your code well enough to make necessary changes. Include appropriate documentation and organize the code in such a way as to make this easy. Remember the golden rule!

If your program is written in Java, make maximum use of javadoc comments for internal documentation, and include appropriate javadoc-generated .html files with your source code.

Test Plan and Test Report

The test plan should follow the outline on page 478 of the text by Bruegge. Each test case should be documented by a test case specification as outlined on page 480. The test report should incorporate a Test Incident Report and a Test Summary Report as discussed on page 480.

APPENDIX A - SENIOR PROJECT PROPOSAL FORM

Student Name(s) _____

COMPUTER SCIENCE SENIOR PROJECT PROPOSAL

The following statement appears in the Gordon college catalog with regard to the Computer Science major - General Concentration: "Students must carry out a senior project (approved in advance by the department) in which they demonstrate the ability to apply classroom learning to an actual computer application or research project of significant size."

A copy of this form, together with necessary project approval signatures and the other information requested, must be submitted to the department before significant work on the proposed project is begun. You are encouraged to submit your proposal as far in advance as possible. Proposals submitted less than one month before the last day of classes for the term immediately preceding your final term before graduation will NOT be considered. You will be notified of the department's acceptance or non-acceptance of this proposal within one working week of submitting it. All work for the project must be complete and necessary documentation must be submitted at least two weeks before the last day of classes of your final term before graduation, so as to allow time for evaluation of your work (and correction of deficiencies if necessary) before you graduate. A copy of this form with the project completion signature of the client must be turned in at that time - the departmental representative will sign this copy after reviewing it.

Date proposal submitted: _____ Date you plan to graduate: _____

Due date for completion of work and submission of all documentation: _____

Title of proposed project: _____

_____

Will this work be done for course credit? _____ If yes, give course number(s) and

academic term(s) under which credit will be earned: _____

Estimate the total number of hours you will spend on this project _____

Person/organization for whom project is being done: _____

_____

Full name, address, and title of the person who will be the client for your project.

_____

_____

I am willing to have this student's work be submitted to Gordon as part of his/her degree requirements. I believe that the work as proposed can realistically be completed within the time frame specified.

_____ _____
(Signature of client) (Date)

Gordon faculty member who will oversee your work on behalf of the department:

_____ _____
(Signature of departmental representative) (Date)

Please attach the following:

1. Summary of what you propose to do and criteria for acceptance of your work. Remember, this will be used as the basis for determining whether your work is acceptable for graduation, so be specific and realistic
2. A draft software project management plan, including a schedule of tasks to be accomplished with milestone dates for each. For each task, estimate the number of hours of effort needed on your part(s).

Departmental action:

____ Approved ____ Conditionally approved subject to conditions listed below ____ Disapproved for reasons listed below

_____ _____
(Signature) (Date)

Completion of project: The work required for this project has been satisfactorily completed

_____ _____
(Signature of client) (Date)

_____ _____
(Signature of departmental representative) (Date)

Note: when this form is initially approved, you will be given the signed original and one copy The copy is for your client. The signed original - with the completion signature of the client added - goes in your final project notebook to be turned in just before graduation - so treat it with care!

APPENDIX B - SAMPLE DOCUMENT TITLE PAGE

(Document title - e.g. Project Plan) **for**

(project title)

To be submitted to the Department of Mathematics and Computer Science,

Gordon College

in partial fulfillment of the requirements for the degree of

Bachelor of Science in Computer Science

by

(your name(s))

Revision date: *(date of this version)*

Document accepted on _____ by _____
(date) (client)

Document accepted on _____ by _____
(date) (departmental representative)

(NOTE: Client signature line is omitted on documents (6) - (9))

APPENDIX C - SOME SUGGESTIONS REGARDING THE SENIOR PROJECT SYSTEM DESIGN DOCUMENT

The Senior Project System Design Document serves as a bridge between the two semesters of CS491-492. The first semester focusses on analysis and high-level design - i.e. identifying what the overall goal of the project is, and specifying the major pieces that need to be built. The second semester focusses on detailed design, implementation and testing - i.e. building these pieces and making sure that, together, they accomplish the original aim of the project. The system design document bridges the two semesters by identifying the pieces that need to be built, and their relationship to one another, in such a way as to allow you to begin working immediately in January on actually building the system (without further need to think about high-level design issues).

Typically, the system design document will center around one or a few diagrams that identify the key pieces that need to be built and their relationship to one another. The diagram(s) will typically be supplemented by appropriate textual discussion. The exact diagram(s) to be produced and the nature of the discussion will vary widely from project to project, depending on the type of project it is, but may include one or more of the following:

- If the software is basically object-oriented, then a class diagram is very appropriate, at least showing the major entity classes for the system.
  - The major pieces that need to be built are, of course, the classes that appear in this diagram.
  - The class diagram shows their relationship to one another, and needs to make careful use of UML symbols for generalization, realization, association (including multiplicity and navigability and possibly role names) (Dependency is not as useful to show).
  - It will probably need to be supplemented with some sort of descriptions of the classes that need to be built: at least a purpose statement for each class, and possibly a list of responsibilities for each.

    An additional level of detail that may prove useful is sequence diagrams showing how the various use cases are realized by messages between objects that are instances of this class - however, care is needed to avoid excessive detail at this point.

- If the software is basically a web-based system, then a statechart is very appropriate.Each state corresponds to a particular page that the user is looking at (and ultimately to the process that creates that page if dynamic html is being used). In some cases a page might need to be represented by more than one state if the state of the page is affected by user actions on the page - e..g a completely filled-out form may allow options that a blank form does not..
  - The transitions between states correspond to links - i.e. a state has a transtion to another state just when the page it represents contains a link to the other page.
  - The transitions are annoted with information about the parameters that are passed as part of that link

- If the software involves a relational database, then a schema diagram for the database (as is the CPS352 text p. 47) is helpful. It may also be useful to spell out what changes are made to the database corresponding to the different use cases - e.g. “this use case results in inserting a new row in the ___ table”.

APPENDIX D - EVALUATION FORM FOR SENIOR PROJECT NOTEBOOK

CPS492 - SENIOR SEMINAR: SOFTWARE ENGINEERING

Evaluation of initial project notebook submission - include this form with the initial submission of the notebook due on the first Monday after spring break.

Student(s) _____

_____

(Remainder of the form to be filled out by the professor and returned to you)

Issues noted below must be addressed in the final project notebook turned in at the end of the course. In addition, of course, project functionality issues identified during testing must be addressed.

<u>Required Documents reviewed previously</u>

1. Approved senior project proposal and problem statement

____ OK ____ Signature(s) missing ____ Totally missing

Note: at this time, only the initial project approval signatures are required. When the completed notebook is turned in at the end of the semester, this form must have the necessary final completion signature of the client!!

2. Software Project Management Plan

____ OK ____ Signature(s) missing ____ Totally missing

3. Social Impact Statement.

____ OK ____ Signature(s) missing ____ Totally missing

4. Requirements Analysis Document.

____ OK ____ Signature(s) missing ____ Totally missing

5. User's Manual. (Final version - reflecting any changes made during implementation)

____ OK ____ Signature(s) missing ____ Totally missing

6. System Design.

____ OK ____ Signature(s) missing ____ Totally missing

Material not previously reviewed

7. Software installation and maintenance documentation

____ OK ____ Inadequate ____ Totally missing

Comments:

8. Detailed design documents and source code listings of all software components. This should be preceded by an outline showing the hierarchical relationship of the components to one another.

   At this point, it is not expected that the source code will be the final source code, since changes will likely be necessitated as a result of testing. However, it is expected that the code submitted will be properly organized and documented.

____ Outline OK ____ Outline inadequate ____ Outline missing

Comments:

____ Code OK ____ Code inadequately documented _____ Code missing

Comments:

The following is not due until the final submission

8b. Updated source code in both printed and machine-readable form (on a disk or server).

9. Test plan and test report prepared by peer tester.